

# SOCLP: A Set Oriented Calculus for Logic Programming

Rafael Caballero      Y. García-Ruiz  
F. Sáenz-Pérez

TECHNICAL REPORT SIP - 02/2006

Dep. Lenguajes, Sistemas Informáticos y Programación  
Univ. Complutense de Madrid

June 2006

## Abstract

This paper presents SOCLP (Set Oriented Calculus for Logic Programming), a proof calculus for pure Prolog programs with negation. The main difference of this calculus w.r.t. other related approaches is that it deals with the answer set of a goal as a whole, instead of considering each single answer separately. We prove that SOCLP is well defined, in the sense that, at most, one answer set can be derived from a goal, and that the derived set is correct and complete w.r.t. the logical meaning of the program. However, not all the answer sets admit a proof in SOCLP. For this reason, the paper also introduces another calculus,  $\text{SOCLP}_{\rightarrow}$ , which allows to prove finite approximations of (possibly infinite) answer sets.  $\text{SOCLP}_{\rightarrow}$  is coupled, through the inference rule for dealing with negation, with a third calculus  $\text{SOCLP}_{\leftarrow}$ , which proves whether a set is a superset of the answer set of a given goal. The paper also shows how the three calculi are related and their main properties.

# Chapter 1

## Introduction

The semantics of logic programs was firstly studied in the seminal paper [4], which introduced the ubiquitous immediate consequence operator  $T_P$ . In [3], the negation as failure rule was introduced as an effective means of deducing negative information for logic programs, and proposed the completion of a program as a description of its meaning. These and further approaches are based on SLD resolution for logic programming, as proposed by [6], a particular refinement of the resolution principle [9]. The drawback of this point of view is that one cannot directly reason about the meaning of a program in terms of answer sets. In particular, this is an inconvenience when using algorithmic debugging [10] for detecting missing answers, where the complete set of answers for any subgoal of the subcomputation is needed. Also, deductive databases, for which semantics is usually defined in the same way as logic programming [13], may be better described with a set oriented calculus, since the natural answer in this context is a set of tuples, as in relational databases.

This paper presents SOCLP (Set Oriented Calculus for Logic Programming), a proof calculus for pure Prolog programs with negation, which deals with the set of answers of a goal as a whole, instead of considering each single answer separately. However, we will see that the completeness of SOCLP can only be ensured for a restrictive class of logic programs, namely the hierarchical programs [7, 5] over finite Herbrand universes. For this reason, the paper also introduces another calculus,  $\text{SOCLP}_{\rightarrow}$ , which establishes whether a given set is a subset of the (possibly infinite) answer set of a certain goal. Therefore,  $\text{SOCLP}_{\rightarrow}$  allows to prove finite approximations of the answer set.  $\text{SOCLP}_{\rightarrow}$  is coupled, through the inference rule for dealing with negation, with a third calculus  $\text{SOCLP}_{\leftarrow}$ , which proves whether a set is a superset of the answer set of a given goal.

We also include theoretical results proving properties of the three calculi.

In particular, we prove that SOCLP is well defined, in the sense that only one set can be derived from a goal, and that such set represents faithfully the answer set of a goal. We also show how the three calculi are related by proving that a set of answers can be inferred from a given goal in SOCLP iff the set can be inferred from both  $\text{SOCLP}_{\leftarrow}$  and  $\text{SOCLP}_{\rightarrow}$ .

To the best of our knowledge, only few papers in the field of declarative debugging address calculi involving goal answer sets (e.g., [11]). However, these works do not consider programs with negation, and usually represent the sets as disjunctions of logic formulas with variables instead of dealing directly with set of ground terms. While their approach is more useful for representing the answers of Prolog systems, ours is more oriented to the case of deductive databases where the answers are assumed to be sets of ground terms.

The limitation of our approach is that, in general, infinite proofs should be needed for recursive goals with function symbols, and hence it seems no adequate for describing logic programs. But if we consider deductive databases, termination is guaranteed provided some conditions. These conditions can avoid infinite proof trees and restrict both function symbols and negation. This is the case of safe Datalog programs with stratified negation [12], where finiteness and termination is ensured.

Our paper will be organized as follows: First, a motivating section introduces our setting and highlights their advantages. Second, we pose some preliminaries about the logic language we adhere to. The third section presents the set oriented calculus SOCLP intended to represent goal meanings as tuple sets. Section four introduces the two calculi  $\text{SOCLP}_{\leftarrow}$  and  $\text{SOCLP}_{\rightarrow}$  and their properties. Finally, some conclusions and future work are pointed out.

# Chapter 2

## Preliminaries

In this section, we present the notation and definitions used throughout the paper which somehow differ from other approaches to logic programming. For other basic definitions about this paradigm, we refer the reader to [1].

We consider programs with the syntax of pure Prolog programs with negation but without "impure" features. A program  $\mathcal{P}$  is therefore a set of *normal clauses* [7]. In order to distinguish the different clauses defining the same predicate  $p$ , we use subindexes following any arbitrary criterium (such as the textual order in the program). Thus, if a predicate  $p$  is defined through  $m$  clauses we will denote them as  $p_1, \dots, p_m$ . Each normal clause  $p_i$  must have the form:

$$\underbrace{p_i(\bar{t}_n)}_{\text{head}} :- \underbrace{l_1(\bar{a}_{k_1}^1), \dots, l_m(\bar{a}_{k_m}^m)}_{\text{body}}.$$

corresponding to the first order logic formula  $p_i(\bar{t}_n) \leftarrow l_1(\bar{a}_{k_1}^1) \wedge \dots \wedge l_m(\bar{a}_{k_m}^m)$ , where the variables whose first occurrence is on the head of the clause have implicit universal quantifiers and the variables whose first appearance is in the body of the clause have implicit existential quantifiers. The notation  $(\bar{t}_n)$  denotes the  $n$ -ary tuple  $(t_1, \dots, t_n)$ . In particular, we represent by  $()$  the 0-ary tuple, commonly called the *unit* tuple. The symbols  $t_i$  (with  $1 \leq i \leq n$ ) and  $a_v^u$  with  $1 \leq u \leq m$ ,  $1 \leq v \leq k_u$  represent *terms*, defined as usual in logic programming: any variable and constant is a term, and any expression  $f(\bar{t}_n)$  (with a function symbol  $f$  of arity  $n$  and terms  $t_i$  ( $1 \leq i \leq n$ )) is a term as well. The set of all the tuples of  $n$  terms that can be built using the constants and functions of a program  $\mathcal{P}$  is denoted in the rest of the paper as  $U_n$ . Notice that  $U_n$  is infinite if the set of terms (the *Herbrand universe*) is infinite. The symbols  $l_i(\bar{a}_{k_i}^i)$  with  $1 \leq i \leq m$  stand for *literals* which can be either positive atoms of the form  $p(\bar{a}_{k_i}^i)$  or negated atoms  $\neg p(\bar{a}_{k_i}^i)$ . Sometimes, we will also be interested in *extended literals* which can be of the form  $p(\bar{a}_{k_i}^i)$ ,  $p_j(\bar{a}_{k_i}^i)$

<code>topicArea<sub>1</sub>(logic,maths)</code>	<code>: - true.</code>
<code>topicArea<sub>2</sub>(topology,maths)</code>	<code>: - true.</code>
<code>topicArea<sub>3</sub>(logic,computers)</code>	<code>: - true.</code>
<code>main<sub>1</sub>(X)</code>	<code>: - topicArea(X,maths), ¬ topicArea(X,computers).</code>

Figure 2.1: Relating topics to areas of knowledge

and  $\neg p(\bar{a}_{k_i}^i)$ . Including names of clauses as possible (extended) literals is consistent with considering each clause  $p_i$  as a predicate defined by just one clause, and any predicate  $p$  defined by the implicit formula:

$$\forall X_1, \dots, X_n (p(\bar{X}_n) \leftarrow p_1(\bar{X}_n) \vee \dots \vee p_m(\bar{X}_n))$$

where  $n$  is the predicate arity and  $X_j$  is a variable for every  $1 \leq j \leq n$ .

*Goals* will be literals  $l(\bar{t}_n)$  with  $(t_n) \in U_n$  and with all the variables in the goal assumed to be existentially quantified. Although the usual definition of goals in logic programming considers conjunctions of literals, ours has no lack of generality; whenever we need to use a conjunction of literals  $l_1(\bar{a}_{k_1}^1), \dots, l_m(\bar{a}_{k_m}^m)$  as a goal, we replace it by the goal  $main(\bar{X}_n)$ , assuming the existence of a new predicate *main* defined by only one clause of the form:

$$\text{main}_1(\bar{X}_n) :- l_1(\bar{a}_{k_1}^1), \dots, l_m(\bar{a}_{k_m}^m)$$

where  $\{X_1, \dots, X_n\}$  is the set of variables in  $l_1(\bar{a}_{k_1}^1), \dots, l_m(\bar{a}_{k_m}^m)$ .

**Example 1** *Figure 2.1 presents a small program written following the conventions described so far. The predicate `topicArea` relates topics to their area of knowledge. The three clauses of this predicate are facts, which are displayed in our setting by including the special propositional constant `true` as the body of each clause. The main predicate of the example holds for those values  $X$  such that are topics of the field of mathematics but not of the field of computers.*

An *answer* of a positive goal  $p(\bar{t}_n)$  w.r.t. a program  $\mathcal{P}$  is a tuple of terms  $(\bar{a}_n)$  such that:

- i)  $(\bar{a}_n) \in U_n$ .
- ii) There exists a substitution  $\theta$  verifying  $(\bar{t}_n)\theta = \bar{a}_n$ .
- iii)  $l(\bar{a}_n)$  is a logical consequence of the set of logic formulas represented by the program.

We say that  $\theta$  is the *associated substitution* to the answer  $(\bar{a}_n)$ . The first condition limits our possible answers to ground terms. For instance, the only answer of the goal  $main(Y)$  w.r.t. the program of Figure 2.1 is (*topology*).

An *answer* for a negative goal  $\neg p(\bar{t}_n)$  w.r.t. a program  $\mathcal{P}$  is a tuple of terms  $(\bar{a}_n)$  such that  $(\bar{a}_n)$  is not an answer of  $p(\bar{t}_n)$ . We use the expression *answer set* to indicate the set containing all the answers of a given goal. For instance, the answer set of the goal  $\neg main(X)$  is the set  $\{(logic), (maths), (computers)\}$ , since these are the elements of  $U_1$  that are not answers of  $main(X)$ .

Finally, we define a special kind of programs: we say that a program is *hierarchical* [7, 5] if there exists some level mapping such that every clause  $p_i(\bar{t}_n) : -l_1(\bar{a}_{k_1}^1), \dots, l_m(\bar{a}_{k_m}^m)$  verifies that the level of every predicate occurring in the body is less than the level of  $p$ . A *level mapping* of a program is a mapping from its set of predicate symbols to the natural numbers. We call *level* of a predicate to the value of predicate under such mapping. For instance, the program in Figure 2.1 is a hierarchical program because we can define the following mapping:  $\{ true \mapsto 0, topicArea \mapsto 1, main \mapsto 2 \}$ .

# Chapter 3

## The SOCLP Calculus

In this section, we present the proof calculus SOCLP, which allows to prove that a set  $S$  is the answer set of a goal  $G$  w.r.t. a program  $\mathcal{P}$ . We will use the notation  $\mathcal{P} \vdash_{\text{SOCLP}} G \leftrightarrow S$ , with  $G$  a goal and  $S$  a set of ground terms, to indicate that the *statement*  $G \leftrightarrow S$  can be deduced in SOCLP using a finite number of steps. In this case, we will say that  $G \leftrightarrow S$  has a proof in SOCLP, and that  $S$  is the SOCLP-set of  $G$ . The five rules of SOCLP can be seen in Figure 3.1. The first two inference rules correspond to the trivial cases of a goal being either *true* or  $p_i(\bar{a}_n)$  with  $(\bar{a}_n)$  not unifiable with the head of any clause for  $p_i$ . In the first case, the SOCLP-set only contains the unit tuple  $()$ , since we assume that this 0-ary predicate always holds. In the second case, if the most general unifier of  $(\bar{a}_n)$  and the tuple at the head of  $p_i$  does not exist, it is easy to check that the goal has no answer. Thus, the SOCLP-set is  $\emptyset$ . The inference rule (NEG $\leftrightarrow$ ) says that the SOCLP-set of a negated atom  $\neg p(\bar{t}_n)$  is the complementary of the SOCLP-set of the corresponding positive atom w.r.t.  $U_n$ . That is, we use the *closed world assumption* [8], which assumes that all atoms not entailed by a program are false. This point of view is convenient for working with answer sets, which makes it widely used in database logic languages (see for instance [13], chapter 10). (PR $\leftrightarrow$ ) defines the SOCLP-set of a positive atom as the union of the SOCLP-sets obtained by using its defining clauses. Finally, (CL $\leftrightarrow$ ) explains how to obtain the SOCLP-set  $S$  of a clause  $p_i(\bar{t}_n)$  from the SOCLP-sets of the literals of its body. The clause  $(p_i(\bar{a}_n) :- l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m)) \in P$  is assumed to have new variables, different from those in  $(\bar{t}_n)$ . This rule says that all the premises must be affected by the most general unifier of  $(\bar{a}_n)$  and  $(\bar{t}_n)$ . Then, each substitution  $\theta'$  that generalizes the associated substitutions of one element of the SOCLP-set of each body literal produces an element in  $S$ . Conversely, each substitution associated to an element of  $S$  must correspond to the restriction of a more general  $\theta'$  that generalizes

**SOCLP**

(TR $\leftrightarrow$ )  $\frac{}{true \leftrightarrow \{()\}}$

(EMP $\leftrightarrow$ )  $\frac{}{p_i(\bar{t}_n) \leftrightarrow \emptyset}$  if  $(p_i(\bar{a}_n) :- l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m)) \in P$ , and  $m.g.u.((\bar{a}_n), (\bar{t}_n))$  does not exist.

(NEG $\leftrightarrow$ )  $\frac{p(\bar{t}_n) \leftrightarrow S_2}{\neg p(\bar{t}_n) \leftrightarrow S_1}$  if  $S_1 = U_n \setminus S_2$

(PR $\leftrightarrow$ )  $\frac{p_1(\bar{t}_n) \leftrightarrow S_1 \dots p_k(\bar{t}_n) \leftrightarrow S_k}{p(\bar{t}_n) \leftrightarrow S}$  if  $S = \bigcup_{i=1}^k S_i$ , and  $p_1, \dots, p_k$  are all the clauses of  $p$  in  $P$

(CL $\leftrightarrow$ )  $\frac{l_1(\bar{b}_{k_1}^1)\theta \leftrightarrow S_1 \dots l_m(\bar{b}_{k_m}^m)\theta \leftrightarrow S_m}{p_i(\bar{t}_n) \leftrightarrow S}$  if:

- $S \subseteq U_n$
- $(p_i(\bar{a}_n) :- l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m)) \in P$
- $\theta = m.g.u.((\bar{a}_n), (\bar{t}_n))$
- $(\bar{t}_n)\theta\theta' \in S$  for all  $\theta'$  s.t.  $(\bar{b}_{k_i}^i)\theta\theta' \in S_i$  with  $i = 1 \dots m$
- for all  $(\bar{t}'_n) \in S$  exists  $\theta'$  s.t.  $(\bar{t}'_n) = (\bar{t}_n)\theta\theta'$  and  $(\bar{b}_{k_i}^i)\theta\theta' \in S_i$  for each  $i = 1 \dots m$

Figure 3.1: The SOCLP calculus

the associated substitutions of one element of the SOCLP-set of each body literal.

**Example 2** *As an example of inference using SOCLP, Figure 3.2 includes a SOCLP proof tree for  $\mathcal{P} \vdash_{\text{SOCLP}} \text{main}(Y) \leftrightarrow \{(\text{topology})\}$ . The root contains the initial statement, and the children of any node correspond to the premises of the SOCLP inference rule applied at the node. Below the tree are listed the renaming of the clauses and the m.g.u. associated to each application of the  $(CL_{\leftrightarrow})$  inference rule. For instance, the first application of  $(CL_{\leftrightarrow})$  corresponds to the clause for main. The children correspond to the body of the clause after applying the m.g.u.:*

$\text{topicArea}(Y, \text{maths}), \neg \text{topicArea}(Y, \text{computers}))$

*The SOCLP-set for the first literal  $\text{topicArea}(Y, \text{maths})$  is  $\{ \text{logic}, \text{topology} \}$ , corresponding to the substitutions  $\theta' = \{Y \mapsto \text{logic}\}$  and  $\theta'' = \{Y \mapsto \text{topology}\}$ , respectively. The SOCLP-set of the second literal is  $\{ \text{maths}, \text{computers}, \text{topology} \}$ , with associated substitutions  $\theta' = \{Y \mapsto \text{maths}\}$ ,  $\theta'' = \{Y \mapsto \text{computers}\}$ , and  $\theta''' = \{Y \mapsto \text{topology}\}$ . Only the substitution  $\theta' = \{Y \mapsto \text{topology}\}$  corresponds to the SOCLP-sets of both literals, and for that reason topology (underlined in the tree) is the only element in the SOCLP-set for  $\text{main}_1$ , following the requirements of the fourth and fifth conditions of  $(CL_{\leftrightarrow})$ .*

This example shows that in SOCLP neither the order of the literals in the body of a clause nor the textual order of the clauses of the same predicate is important. The following proposition ensures that the calculus defines at most one SOCLP-set for any given goal.

**Proposition 1** *Let  $\mathcal{P}$  be a program,  $l(\bar{t}_n)$  and  $S_a, S_b$  two sets such that  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S_a$  and  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S_b$ . Then  $S_a = S_b$ .*

**Proof.**

Let  $T$  be a proof tree for  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S_a$ . The result can be proved using induction on the height of  $T$ . If  $\text{height}(T) = 0$  (basis), observing the inference rules of the calculus it is straightforward to check that either the inference  $(\text{TR}_{\leftrightarrow})$  or  $(\text{EMP}_{\leftrightarrow})$  has been applied at the tree root. In the first case, the only set  $S$  allowing a proof for the statement  $\text{true} \leftrightarrow S$  is  $\{()\}$ , and therefore  $S_a \equiv S_b \equiv \{()\}$ . In the second case  $(\text{EMP}_{\leftrightarrow})$  has been applied, and  $l(\bar{t}_n)$  must be of the form  $p_i(\bar{t}_n)$  with  $p_i$  a program clause of the form

$$p_i(\bar{a}_n) : -l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m)$$

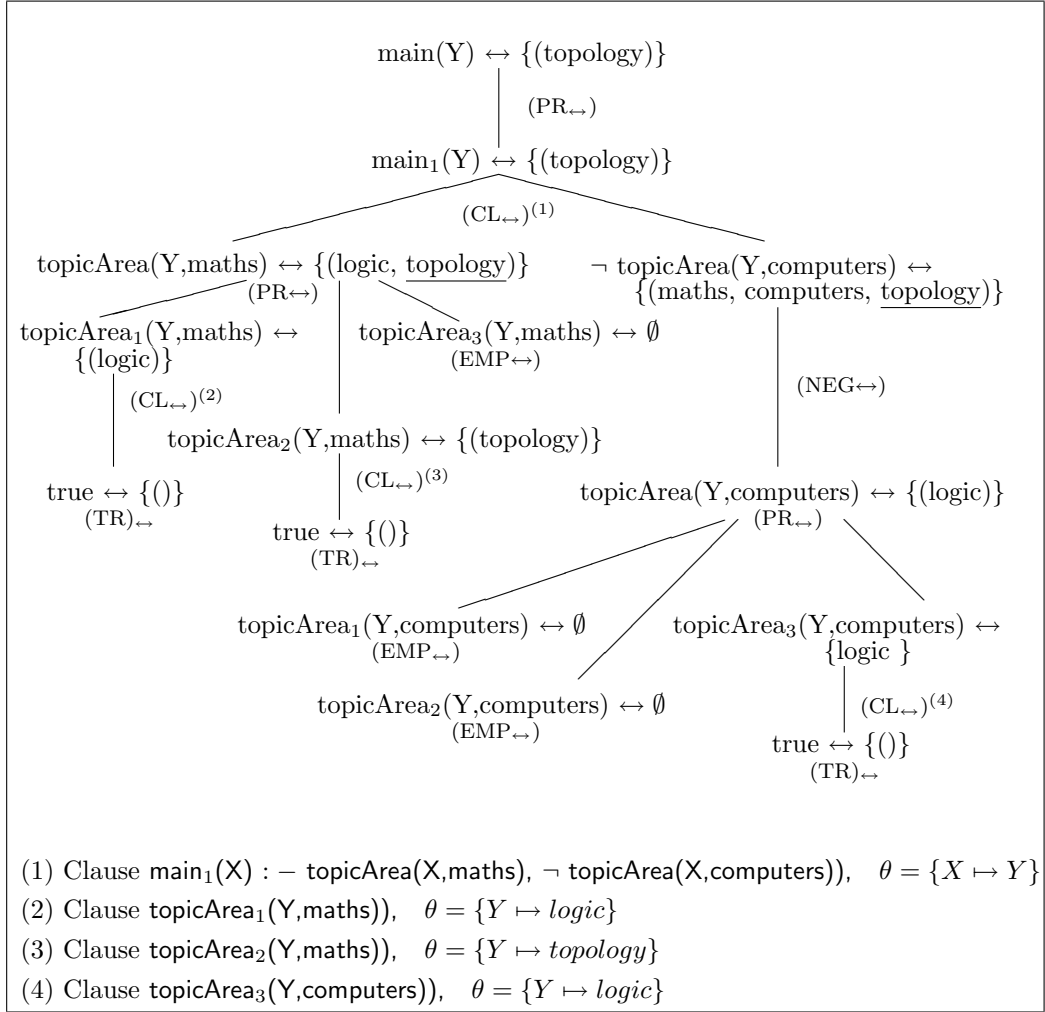


Figure 3.2: Inference using the Calculus  $\text{SOCLP}_{\leftrightarrow}$

and the  $m.g.u.((\bar{a}_n), (\bar{t}_n))$  does not exist. In this case only the  $(EMP_{\leftrightarrow})$  inference rule can be applied, at the root of  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S_b$ , and hence  $S_a \equiv S_b \equiv \emptyset$ .

If  $height(T) > 0$  (inductive step) the inference rule applied at the root of  $T$  can be either  $(NEG_{\leftrightarrow})$ ,  $(PR_{\leftrightarrow})$  or  $(CL_{\leftrightarrow})$ . We next show the proof of the result in the first case. The proof is analogous for the other two rules.

If the inference rule applied at the root of  $T$  is  $(NEG_{\leftrightarrow})$ , then  $l(\bar{t}_n) \equiv \neg p(\bar{t}_n)$  for some predicate  $p$ , and the corresponding inference step must be of the form:

$$\frac{p(\bar{t}_n) \leftrightarrow S_1}{\neg p(\bar{t}_n) \leftrightarrow S_a}$$

with  $S_a = U_n \setminus S_1$ . Any proof of  $\mathcal{P} \vdash_{\text{SOCLP}} \neg p(\bar{t}_n) \leftrightarrow S_b$  must also be of the form

$$\frac{p(\bar{t}_n) \leftrightarrow S_2}{\neg p(\bar{t}_n) \leftrightarrow S_b}$$

with  $S_b = U_n \setminus S_2$ . The proof tree  $T'$  of  $\mathcal{P} \vdash_{\text{SOCLP}} p(\bar{t}_n) \leftrightarrow S_1$  is a children subtree of  $T$ , and therefore  $height(T') < height(T)$ . Applying the induction hypothesis it follows that  $S_1 \equiv S_2$ , and hence  $S_a \equiv S_b$ .  $\square$

The next theorem establishes the relationship between the SOCLP proofs and the logical meaning of the program, proving that the SOCLP-set of any goal is actually its answer set.

**Theorem 1** (*Soundness and weak completeness of SOCLP*)

Let  $\mathcal{P}$  be a program,  $l(\bar{t}_n)$  a goal, and  $S$  a set such that  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S$ . Then  $(\bar{a}_n) \in S$  iff  $(\bar{a}_n)$  is an answer of  $l(\bar{t}_n)$ .

**Proof.** We use induction on the height of a proof tree  $T$  for  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S$ , with  $l$  any extended literal (using extended literals instead of literals is more general and convenient in the proof).

$\Rightarrow$  We start considering the SOCLP inference used at the root of  $T$ :

- $(TR_{\leftrightarrow})$ . In this case  $()$  is the only element in  $S$ , and it is straightforward to check that  $()$  is an answer for *true* w.r.t. any program  $P$ .
- $(EMP_{\leftrightarrow})$ . The result holds trivially since there is no element in  $S$ .

– (NEG<sub>↔</sub>). The inference must be of the form:

$$\frac{p(\bar{t}_n) \leftrightarrow S'}{\neg p(\bar{t}_n) \leftrightarrow S} \quad \text{if } S = U_n \setminus S'$$

Applying the induction hypothesis to the premise, all the elements in  $S'$  are answers of  $p(\bar{t}_n)$ . But  $S = U_n \setminus S'$  which means that any element of  $S$  is not an answer of  $p(\bar{t}_n)$  and therefore is an answer of  $\neg p(\bar{t}_n)$  according to the definition of answer of a negative goal.

– (PR<sub>↔</sub>). The induction hypothesis can be applied to the premises, showing that every element in  $S_i$  is an answer for  $p_i(\bar{t}_n)$  for  $i = 1 \dots k$ .  $S$  is the union of the  $S_i$ , and therefore every  $s \in S$  is an answer for some  $p_i(\bar{t}_n)$ . It is straightforward to check that any answer for  $p_i(\bar{t}_n)$  is as well an answer of  $p(\bar{t}_n)$  because  $p$  is defined as the disjunction of all its clauses  $p_i$ .

– (CL<sub>↔</sub>). Let  $(p_i(\bar{a}_n) :- l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m))$  the renaming of the clause for  $p_i$  used in the inference rule.

Let  $s$  be an element of  $S$ . By the fifth condition of the inference rule conditions,  $s$  is of the form  $(\bar{t}_n)\theta\theta'$ , with  $\theta$  the m.g.u. of  $(\bar{t}_n)$  and  $(\bar{a}_n)$ , where  $p_i(\bar{a}_n)$  is the head of a renaming of the clause for  $p_i$ , and  $\theta'$  is such that  $(\bar{b}_{k_i}^i)\theta\theta' \in S_i$  with  $i = 1 \dots m$ . By the induction hypothesis the  $(\bar{b}_{k_i}^i)\theta\theta'$  are answers of  $l(\bar{b}_{k_i}^i)\theta$ . This means that all the premises of the clause are satisfied by  $\theta\theta'$ , and hence that  $(\bar{t}_n)\theta\theta'$ , i.e.  $s$ , is an answer of  $p_i(\bar{t}_n)\theta$ . Finally, it is easy to check that if  $(\bar{t}_n)\theta\theta'$  is an answer of  $p_i(\bar{t}_n)\theta$ , then  $(\bar{t}_n)(\theta\theta')$  is an answer of  $p_i(\bar{t}_n)$ .

⇐ – (TR<sub>↔</sub>). Then  $l(\bar{t}_n)$  is *true()*, or in short simply *true*. And the only possible answer for *true* is  $()$ , since this is the only element in  $U_0$  and *true* is a logical consequence of any program. And  $()$  is in  $S$ , as the theorem claims.

– (EMP<sub>↔</sub>). In this case  $l(\bar{t}_n)$  is  $p_i(\bar{t}_n)$ . The non existence of the most general unifier implies that there is no  $\theta$  such that  $p_i(\bar{t}_n)\theta$  is a logical consequence of the program, and therefore the answer set is  $\emptyset$  and the result holds.

– (NEG<sub>↔</sub>). The inference must be of the form:

$$\frac{p(\bar{t}_n) \leftrightarrow S'}{\neg p(\bar{t}_n) \leftrightarrow S} \quad \text{if } S = U_n \setminus S'$$

Applying induction to the premise, all the answers of  $p(\bar{t}_n)$  are elements of  $S'$ . But  $S = U_n \setminus S'$ , and hence every answer of  $\neg p(\bar{t}_n)$  is in  $S$ , due to the definition of answers for negative goals.

- (PR $\leftrightarrow$ ). The predicate  $p$  is defined as the disjunction of its clauses  $p_1, \dots, p_k$ . Then any answer  $s$  of  $p(\bar{t}_n)$  must be the answer of some  $p_i(\bar{t}_n)$ , with  $1 \leq i \leq k$ . Applying the induction hypothesis to the  $i$ -th premise of the inference rule, we have that  $s \in S_i$ , and hence  $s \in S$  ( $S$  is the union of all the  $S_i$ ).
- (CL $\leftrightarrow$ ). Let  $(p_i(\bar{a}_n) :- l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m))$  the renaming of the clause for  $p_i$  used in the inference rule. Let  $s \equiv (\bar{t}_n)\sigma_1$  be an answer for  $p_i(\bar{t}_n)$ . By the definition of answer,  $s$  is a logical consequence of the program. Since  $p_i$  is defined by only one clause, this means that there must exist a substitution  $\sigma_2$  such that  $(p_i(\bar{a}_n) \leftarrow l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m))\sigma_2$  can be deduced from the program, with  $s \equiv (\bar{a}_n)\sigma_2$ . Without loss of generality we can assume that  $dom(\sigma_1) \cap dom(\sigma_2) = \emptyset$  (the instance of clause is a renaming such that it has no common variable with  $(\bar{t}_n)$ ). Then the substitution  $\sigma =_{def.} \sigma_1 \cup \sigma_2$  verifies  $s = (\bar{t}_n)\sigma = (\bar{a}_n)\sigma$ , i.e.  $\sigma$  is a unifier of  $(\bar{t}_n)$  and  $(\bar{a}_n)$ , and by the definition of most general unifier,  $\sigma = \theta\theta'$  with  $\theta$  the m.g.u. of  $(\bar{t}_n)$  and  $(\bar{a}_n)$ , and  $\theta'$  some substitution.

Now, using the induction hypothesis in the premises of the inference rule we have that all the answers of  $l(\bar{b}_{k_i}^i)\theta$  with  $i = 1 \dots k$  are in the corresponding set  $S_i$ . Since  $\sigma$  is more particular than  $\theta$ , all the answers of  $l(\bar{b}_{k_i}^i)\sigma$  will be also in  $S_i$  (remember that the answers are ground terms), i.e. all the answers of  $l(\bar{b}_{k_i}^i)\theta\theta'$  will be in  $S_i$  or, equivalently,  $\theta'$  is the associated substitution to an answer of  $l(\bar{b}_{k_i}^i)\theta$  for each  $i = 1 \dots k$ . By the fourth condition of the inference rule (CL $\leftrightarrow$ ),  $(\bar{t}_n)\theta\theta'$  must be in  $S$ , which means that  $(\bar{t}_n)\sigma \in S$ , and since  $\sigma = \sigma_1 \cup \sigma_2$  and  $dom(\sigma_2) \cap var(\bar{t}_n) = \emptyset$ ,  $(\bar{t}_n)\sigma_1 \in S$ , i.e.  $s \in S$ .

□

The theorem states that, whenever a SOCLP-proof for  $l(\bar{t}_n) \leftrightarrow S$  exists, we can ensure that  $S$  is precisely the answer set of  $l(\bar{t}_n)$ . Hence, we can trust the SOCLP proofs as suitable descriptions of the answer set of a goal. However, not all the answer sets admit a SOCLP proof tree.

**Example 3** Consider, for instance, the small program:

$$\begin{aligned} p_1(a) &: - \text{true.} \\ p_2(X) &: - p(X). \end{aligned}$$

It is easy to check out that the answer set of the goal  $p(X)$  is  $\{(a)\}$ . However, the statement  $p(X) \leftrightarrow \{(a)\}$  cannot be proved in SOCLP.

The next proposition establishes a set of programs w.r.t. all the answer sets of a goal admit a SOCLP proof:

**Proposition 2** (*Completeness of hierarchical programs over finite Herbrand universes*)

Let  $\mathcal{P}$  be a hierarchical program over a finite Herbrand universe and  $l(\bar{t}_n)$  a goal. Then, there exists some set  $S$  such that  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S$ .

**Proof.** We prove by complete induction the next, more general, result: For every natural number  $m$ , every extended literal  $l(\bar{t}_n)$  containing a predicate of level  $m$  has a set  $S$  such that  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S$ .

- **Basis.** The basis corresponds to the predicate with minimum level, which is necessarily the special symbol *true*. Indeed, it is easy to check that every hierarchical program must contain a clause with body *true*, and that this is the only predicate which is not defined in terms of other program predicates. In this case a proof of the form  $\mathcal{P} \vdash_{\text{SOCLP}} \text{true} \leftrightarrow S$  exists with  $S = \{()\}$ , because such statement can be always proved by using the SOCLP inference rule (TR $\leftrightarrow$ ).
- **Inductive case.** Given  $r > 0$ , we assume that the result holds for all natural numbers less than  $r$ . We distinguish cases depending on the form of  $l(\bar{t}_n)$ :
  - $l(\bar{t}_n)$  is of the form  $p(\bar{t}_n)$ . Every  $p_i$  with  $i = 1 \dots k$  ( $k$  the number of defining clauses of  $p$ ), has less level than  $p$ , which means by induction hypothesis that a proof of the form  $\mathcal{P} \vdash_{\text{SOCLP}} p_i(\bar{t}_n) \leftrightarrow S_i$  exists. Then by defining  $S$  as  $\cup_{i=1}^k S_i$  we can prove  $\mathcal{P} \vdash_{\text{SOCLP}} p(\bar{t}_n) \leftrightarrow S$  starting with an (PR $\leftrightarrow$ ) inference rule.
  - $l(\bar{t}_n)$  is of the form  $\neg p(\bar{t}_n)$ . By the previous point we have that there exists an  $S$  such that  $\mathcal{P} \vdash_{\text{SOCLP}} p(\bar{t}_n) \leftrightarrow S'$ . Then by defining  $S$  as  $U_n \setminus S'$  we have that  $\mathcal{P} \vdash_{\text{SOCLP}} \neg p(\bar{t}_n) \leftrightarrow S$  has a proof which starts with the (NEG $\leftrightarrow$ ) inference.
  - $l(\bar{t}_n)$  is of the form  $p_i(\bar{t}_n)$ . If the *m.g.u.*(( $\bar{a}_n$ ), ( $\bar{t}_n$ )) does not exist with  $(p_i(\bar{a}_n) :- l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m))$  a renaming of the clause for  $p_i$ , then  $\mathcal{P} \vdash_{\text{SOCLP}} p_i(\bar{t}_n) \leftrightarrow S$  admits a proof in SOCLP taking  $S$  as  $\emptyset$  and (EMP $\leftrightarrow$ ) the only inference rule needed. Otherwise, if  $\theta = \text{m.g.u.}((\bar{a}_n), (\bar{t}_n))$  exists, then by the induction hypothesis there exists also sets  $S_i$  with  $i = 1 \dots m$  such that

there is a proof for  $\mathcal{P} \vdash_{\text{SOCLP}} l_i(\bar{b}_{k_i}^i)\theta \leftrightarrow S_i$ . Therefore we can construct  $S$  as

$$S = \{(\bar{t}_n)\theta\theta' \mid \theta' \text{ s.t. } (\bar{b}_{k_i}^i)\theta\theta' \in S_i \text{ with } i = 1 \dots m\}$$

and prove  $\mathcal{P} \vdash_{\text{SOCLP}} p_i(\bar{t}_n) \leftrightarrow S$  starting with a  $(\text{CL}_{\leftrightarrow})$  inference rule. Notice that here the premise of a finite Herbrand Universe is important: it ensures that the number of possible substitutions  $\theta'$  is finite and thus that  $S$  can be effectively constructed.

□

By Proposition 1, the set  $S$  is unique, and, by Theorem 1, this set is the answer set of the goal. Thus, the SOCLP calculus defines correctly the semantics of hierarchical programs [5] over finite Herbrand universes from the point of view of the answer sets. Although the proposition provides a rather restrictive set of programs for which SOCLP is complete, this does not mean that SOCLP cannot be applied to non-hierarchical programs.

**Example 4** *The predicate `main` of the following program defines the set of natural numbers which are less than one (i.e., only the number zero):*

$$\begin{aligned} \text{less}_1(\text{zero}, \text{suc}(Y)) &: - \text{true}. \\ \text{less}_2(\text{suc}(X), \text{suc}(Y)) &: - \text{less}(X, Y). \\ \text{main}_1(X) &: - \text{less}(X, \text{suc}(\text{zero})). \end{aligned}$$

*This program is not hierarchical, due to the second rule for `less`, and its Herbrand Universe is infinite. However, as Figure 3.3 shows, using SOCLP is possible to prove that  $\mathcal{P} \vdash_{\text{SOCLP}} \neg \text{main}(X) \leftrightarrow \{(suc(\text{zero})), (suc(suc(\text{zero}))), \dots\}$ , i.e., that all the natural numbers different from zero are not less than `suc(zero)`.*

In the previous example, it is also interesting to note that SOCLP proofs are not always limited to finite answer sets. This is due to the rule for negation, which can convert the proof of an infinite answer set in the proof of a finite answer set, and vice versa (as the first inference step of Figure 3.3 shows). This rule also makes the set of provable statements different from those of pure Prolog programs with negation. For instance, the previous goal  $\neg \text{main}(X)$  has no solution using negation as failure because  $\text{main}(X)$  does not fail (it succeeds with  $X \mapsto \text{zero}$ ).



# Chapter 4

## The SOCLP $\leftarrow$ and SOCLP $\rightarrow$ Calculi

The previous section presented some examples of SOCLP proofs for finite and, even, infinite answer sets. It also showed that it is very easy to find programs and goals whose answer set does not admit a SOCLP proof. Consider for instance the following program:

**Example 5** *A predicate  $\text{nat}$  defining the natural numbers using Peano's axioms:*

$\text{nat}_1(\text{zero}) \quad : - \text{true}.$   
 $\text{nat}_2(\text{suc}(X)) \quad : - \text{nat}(X).$

The answer set of the goal  $\text{nat}(N)$  is the infinite set of program terms representing the set of the natural numbers  $\{ (\text{zero}), (\text{suc}(\text{zero})), (\text{suc}(\text{suc}(\text{zero}))), \dots \}$ . The SOCLP calculus cannot build a proof for this set because of the recursive definition of  $\text{nat}_2$ .

Figure 4.1 introduces two new calculi, SOCLP $\leftarrow$  and SOCLP $\rightarrow$ , that can substitute SOCLP when we are interested in proving that a given set is included in the answer set of a given goal. The two calculi are coupled through the inference rule for negation. Given a goal  $G$  and a set of ground terms  $S$ , we use the notation  $\mathcal{P} \vdash_{\text{SOCLP}\leftarrow} G \leftarrow S$  to indicate that there exists a finite inference for  $G \leftarrow S$  using the calculus SOCLP $\leftarrow$ , and  $\mathcal{P} \vdash_{\text{SOCLP}\rightarrow} G \rightarrow S$  to indicate that there exists a finite inference for  $G \rightarrow S$  using the calculus SOCLP $\rightarrow$ .

The purpose of the calculi is to prove partial approximations to the answer set of a given goal. Their inference rules are similar to the inference rules for SOCLP, but "relaxing" the associated conditions, which increase the set of provable statements. Considering again the Example 5, now it is possible

<u>SOCLP<math>\leftarrow</math></u>	<u>SOCLP<math>\rightarrow</math></u>
(TR $\leftarrow$ ) $\frac{}{true \leftarrow \{()\}}$	(TR $\rightarrow$ ) $\frac{}{true \rightarrow \{()\}}$
(EMP $\leftarrow$ ) $\frac{}{p_i(t_n) \leftarrow \emptyset}$	(UNI $\rightarrow$ ) $\frac{}{p_i(t_n) \rightarrow S}$ if: - $S \subseteq U_n$ - $(p_i(\bar{a}_n) :- l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m)) \in P$ , and $m.g.u.((\bar{a}_n), (\bar{t}_n))$ does not exist
(NEG $\leftarrow$ ) $\frac{p(\bar{t}_n) \rightarrow S_2}{\neg p(t_n) \leftarrow S_1}$ if $S_1 \subseteq U_n \setminus S_2$	(NEG $\rightarrow$ ) $\frac{p(\bar{t}_n) \leftarrow S_2}{\neg p(t_n) \rightarrow S_1}$ if $S_1 = U_n \setminus S_2$
(PR $\leftarrow$ ) $\frac{p_1(\bar{t}_n) \leftarrow S_1 \dots p_k(\bar{t}_n) \leftarrow S_k}{p(t_n) \leftarrow S}$ if $S \subseteq \bigcup_{i=1 \dots k} S_i$	(PR $\rightarrow$ ) $\frac{p_1(\bar{t}_n) \rightarrow S_1 \dots p_k(\bar{t}_n) \rightarrow S_k}{p(t_n) \rightarrow S}$ if $\bigcup_{i=1 \dots k} S_i \subseteq S$
(CL $\leftarrow$ ) $\frac{l_1(\bar{b}_{k_1}^1)\theta \leftarrow S_1 \dots l_m(\bar{b}_{k_m}^m)\theta \leftarrow S_m}{p_i(t_n) \leftarrow S}$ if: - $S \subseteq U_n$ - $(p_i(\bar{a}_n) :- l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m)) \in P$ - $\theta = m.g.u.((\bar{a}_n), (\bar{t}_n))$ - For all $(\bar{t}'_n) \in S$ , exists $\theta'$ s.t. $(\bar{t}'_n) = (\bar{t}_n)\theta\theta'$ and $(\bar{b}_{k_i}^i)\theta\theta' \in S_i$ for $i = 1 \dots m$	(CL $\rightarrow$ ) $\frac{l_1(\bar{b}_{k_1}^1)\theta \rightarrow S_1 \dots l_m(\bar{b}_{k_m}^m)\theta \rightarrow S_m}{p_i(t_n) \rightarrow S}$ if: - $S \subseteq U_n$ - $(p_i(\bar{a}_n) :- l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m)) \in P$ - $\theta = m.g.u.((\bar{a}_n), (\bar{t}_n))$ - $(\bar{t}_n)\theta\theta'$ is in $S$ for all $\theta'$ s.t. $(\bar{b}_{k_i}^i)\theta\theta' \in S_i$ for $i = 1 \dots m$

Figure 4.1: The SOCLP $\leftarrow$  and SOCLP $\rightarrow$  Calculi

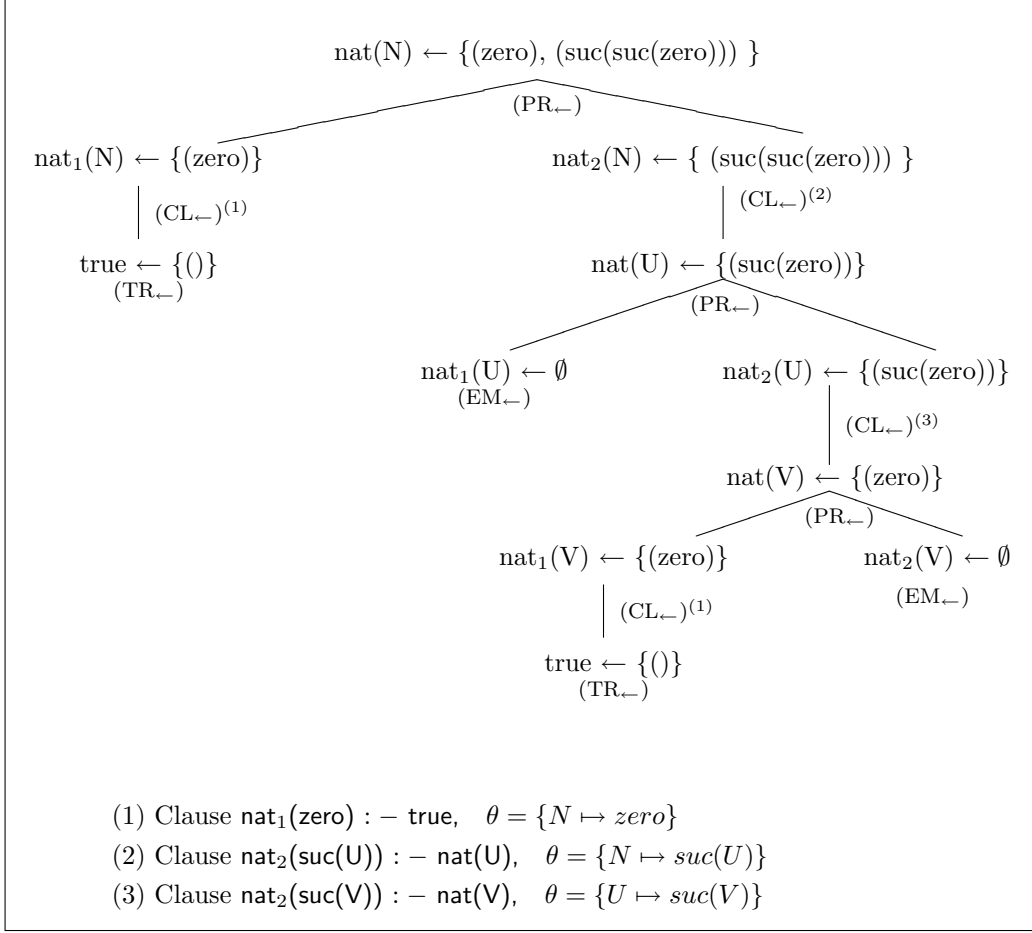


Figure 4.2: Inference using the calculus  $\text{SOCLP}_{\leftarrow}$

to prove that  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} \text{nat}(N) \rightarrow S$ , with  $S$  any finite approximation of the answer set  $\{ \text{zero}, \text{suc}(\text{zero}), \text{suc}(\text{suc}(\text{zero})), \dots \}$ .

For instance, Figure 4.2 shows the proof tree proving that  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} \text{nat}(N) \leftarrow \{(zero), (suc(suc(zero)))\}$ . The relationship between the two calculi and  $\text{SOCLP}$  is stated in the next proposition:

**Proposition 3** *Let  $\mathcal{P}$  be a program,  $l(\bar{t}_n)$  a literal, and  $S$  a set of tuples of arity  $n$ . Then,  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S$  implies  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \rightarrow S$  and  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \leftarrow S$ .*

**Proof.**

We prove that  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S$  implies  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \rightarrow S$  and  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \leftarrow S$ , with  $l(\bar{t}_n)$  any extended literal. The proof proceeds by induction on the height of a proof tree  $T$  for  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S$ .

*Basis* ( $height(T) = 0$ ). Observing the inference rules of the calculus, it is straightforward to check that there are only two alternatives:  $(TR_{\leftrightarrow})$  and  $(EMP_{\leftrightarrow})$ :

- $(TR_{\leftrightarrow})$ . Then  $l(\bar{t}_n)$  is *true* and  $S = \{()\}$ . In this case the statement  $true \leftarrow \{()\}$  can be proved in  $SOCLP_{\leftarrow}$  using the inference rule  $(TR_{\leftarrow})$ , and  $true \rightarrow \{()\}$  can be proved in  $SOCLP_{\rightarrow}$  using the inference rule  $(TR_{\rightarrow})$ .
- $(EMP_{\leftrightarrow})$ .  $l(\bar{t}_n)$  must be of the form  $p_i(\bar{t}_n)$  for some clause  $p_i$ , and  $S = \emptyset$ . Then we can use  $(EMP_{\leftarrow})$  to prove  $p_i(\bar{t}_n) \leftarrow \emptyset$  in  $SOCLP_{\leftarrow}$ , and  $(UNI_{\rightarrow})$  to prove  $p_i(\bar{t}_n) \rightarrow \emptyset$  in  $SOCLP_{\rightarrow}$ , which can be applied because  $\emptyset \subseteq U_n$  and because the rule  $(EMP_{\leftrightarrow})$  ensures that there is no matching rule for  $p_i(\bar{t}_n)$  in  $\mathcal{P}$ .

*Induction* ( $height(T) > 0$ ). The inference rule applied at the root of  $T$  can be either  $(NEG_{\leftrightarrow})$ ,  $(PR_{\leftrightarrow})$  or  $(CL_{\leftrightarrow})$ .

- $(NEG_{\leftrightarrow})$ . Then  $l(\bar{t}_n)$  is  $\neg p(\bar{t}_n)$  for some predicate  $p$ , and the corresponding inference step must be of the form:  $\frac{p(\bar{t}_n) \leftrightarrow S'}{\neg p(\bar{t}_n) \leftrightarrow S}$  with  $S = U_n \setminus S'$ . The proof tree  $T'$  of  $p(\bar{t}_n) \leftrightarrow S'$  verifies  $height(T') < height(T)$ , and applying the induction hypothesis we have that it is possible to prove  $p(\bar{t}_n) \leftarrow S'$  and  $p(\bar{t}_n) \rightarrow S'$  in  $SOCLP_{\leftarrow}$ ,  $SOCLP_{\rightarrow}$ , respectively. Then it is also possible to prove  $\neg p(\bar{t}_n) \leftarrow S$  in  $SOCLP_{\leftarrow}$  by extending the proof of  $p(\bar{t}_n) \leftarrow S'$  with a last  $(NEG_{\leftarrow})$  inference step:

$$\frac{p(\bar{t}_n) \rightarrow S'}{\neg p(\bar{t}_n) \leftarrow S}$$

The condition  $S \subseteq U_n \setminus S'$  required by the rule holds because  $(NEG_{\leftrightarrow})$  implies that  $S = U_n \setminus S'$ . Analogously, it is possible to extend the  $SOCLP_{\rightarrow}$  proof of  $p(\bar{t}_n) \rightarrow S'$  by using the  $(NEG_{\rightarrow})$  inference rule in order to prove  $\neg p(\bar{t}_n) \rightarrow S$ :

$$\frac{p(\bar{t}_n) \leftarrow S'}{\neg p(\bar{t}_n) \rightarrow S}$$

since we know by the conditions of  $(NEG_{\leftrightarrow})$  that  $S = U_n \setminus S'$ .

- $(PR_{\leftrightarrow})$ . The proof is analogous to the previous case and we skip the details.  $l(\bar{t}_n)$  must be of the form  $p(\bar{t}_n)$  for some predicate  $p$ , and the corresponding inference step must be of the form:

$$\frac{p_1(\bar{t}_n) \leftrightarrow S_1 \dots p_k(\bar{t}_n) \leftrightarrow S_k}{p(\bar{t}_n) \leftrightarrow S}$$

with  $S = \bigcup_{i=1}^k S_i$ , and where  $p_1, \dots, p_k$  are all the clauses of  $p$  in  $P$ .

Applying the induction hypothesis to the SOCLP proofs of the premises we have that  $\mathcal{P} \vdash_{\text{SOCLP}_{\rightarrow}} p_i(\bar{t}_n) \rightarrow S_i$ ,  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} p_i(\bar{t}_n) \leftarrow S_i$  for  $i = 1 \dots k$ . Then it is possible to use to proofs of the  $p_i(\bar{t}_n) \leftarrow S_i$  to build a SOCLP $_{\leftarrow}$  proof for  $p(\bar{t}_n) \leftarrow S$  which ends with an application of the (PR $_{\leftarrow}$ ) inference rule:

$$\frac{p_1(\bar{t}_n) \leftarrow S_1 \dots p_k(\bar{t}_n) \leftarrow S_k}{p(\bar{t}_n) \leftarrow S}$$

where the condition  $S \subseteq \bigcup_{i=1}^k S_i$  holds because the inference rule (PR $_{\leftrightarrow}$ ) ensures that  $S = \bigcup_{i=1}^k S_i$ . Analogously, in the case of the SOCLP $_{\rightarrow}$  proof of  $p(\bar{t}_n) \rightarrow S$ :

$$\frac{p_1(\bar{t}_n) \rightarrow S_1 \dots p_k(\bar{t}_n) \rightarrow S_k}{p(\bar{t}_n) \rightarrow S}$$

where the conditions required for applying the (PR $_{\rightarrow}$ ) inference rule are entailed by the conditions of (PR $_{\leftrightarrow}$ ).

- (CL $_{\leftrightarrow}$ ).  $l(\bar{t}_n)$  is  $p_i(\bar{t}_n)$  for some clause  $p_i$  defining a predicate  $p$ , and the corresponding inference step must be of the form:

$$\frac{l_1(\bar{b}_{k_1}^1)\theta \leftrightarrow S_1 \dots l_m(\bar{b}_{k_m}^m)\theta \leftrightarrow S_m}{p_i(\bar{t}_n) \leftrightarrow S}$$

By the induction hypothesis  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l_i(\bar{b}_{k_i}^i)\theta \leftarrow S_i$  for  $i = 1 \dots m$ . Combining the proofs for these statements it is possible to prove  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} p_i(\bar{t}_n) \leftarrow S$  ending with an inference step of the form:

$$(\text{CL}_{\leftarrow}) \frac{l_1(\bar{b}_{k_1}^1)\theta \leftarrow S_1 \dots l_m(\bar{b}_{k_m}^m)\theta \leftarrow S_m}{p_i(\bar{t}_n) \leftarrow S}$$

where the conditions required for applying (CL $_{\leftarrow}$ ) hold since they are a subset of the conditions that were necessary for applying (CL $_{\leftrightarrow}$ ). Analogously, in the case of SOCLP $_{\rightarrow}$  it is possible to combine the proofs of  $l_i(\bar{b}_{k_i}^i)\theta \rightarrow S_i$  by means of the (CL $_{\rightarrow}$ ) inference rule for proving  $p_i(\bar{t}_n) \rightarrow S$ :

$$\frac{l_1(\bar{b}_{k_1}^1)\theta \rightarrow S_1 \dots l_m(\bar{b}_{k_m}^m)\theta \rightarrow S_m}{p_i(\bar{t}_n) \rightarrow S''}$$

Again the conditions required for applying the rules are a subset of the conditions required by (CL $_{\leftrightarrow}$ ) and hence are satisfied.

□

That is, the existence of a SOCLP proof for  $l(\bar{t}_n) \leftrightarrow S$  determines the existence of both a  $\text{SOCLP}_{\leftarrow}$  proof for  $l(\bar{t}_n) \leftarrow S$ , and a  $\text{SOCLP}_{\rightarrow}$  proof for  $l(\bar{t}_n) \rightarrow S$ . But the main goal of  $\text{SOCLP}_{\leftarrow}$  and  $\text{SOCLP}_{\rightarrow}$  is not proving the answer set of a goal, but approximations to this answer set. The next proposition states that the two calculi are also useful for this purpose:

**Proposition 4** *Let  $l(\bar{t}_n)$  be a literal and  $S$  a set such that  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S$ . Then, the two following statements hold:*

- i)  $S' \subseteq S$  for every  $S'$  such that  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \leftarrow S'$*
- ii)  $S' \supseteq S$  for every  $S'$  such that  $\mathcal{P} \vdash_{\text{SOCLP}_{\rightarrow}} l(\bar{t}_n) \rightarrow S'$*

**Proof.**

We check the validity of the two items at the same time by proving the following, more general result:

*Let  $l(\bar{t}_n)$  be a literal and  $S$  a set such that  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S$ . Let  $S'$  be a set such that either  $\mathcal{P} \vdash_{\text{SOCLP}_{\rightarrow}} l(\bar{t}_n) \rightarrow S'$  or  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \leftarrow S'$ . Then, either  $S \subseteq S'$  (if  $\mathcal{P} \vdash_{\text{SOCLP}_{\rightarrow}} l(\bar{t}_n) \rightarrow S'$ ) or  $S' \subseteq S$  (if  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \leftarrow S'$ ).*

Let  $T$  be a proof tree of either  $\mathcal{P} \vdash_{\text{SOCLP}_{\rightarrow}} l(\bar{t}_n) \rightarrow S'$  or  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \leftarrow S'$ . We prove the result using induction on  $\text{height}(T)$ .

*Basis* ( $\text{height}(T)=0$ ). The possible inference rules applied at the root of  $T$  can be either  $(\text{TR}_{\rightarrow})$  ( $\text{TR}_{\leftarrow}$ ),  $(\text{EMP}_{\leftarrow})$  or  $(\text{UNI}_{\rightarrow})$ . We study each possibility:

- $(\text{TR}_{\rightarrow})$  or  $(\text{TR}_{\leftarrow})$ . Then  $l(\bar{t}_n)$  is *true*, and the inference rule applied in the first step for proving  $\mathcal{P} \vdash_{\text{SOCLP}} \text{true} \leftrightarrow S$  is necessarily  $(\text{TR}_{\leftrightarrow})$ . Then both  $S'$  and  $S$  are  $\{()\}$  and the result holds.
- $(\text{UNI}_{\rightarrow})$ . Then  $S' = U_n$  and  $S \subseteq S'$  trivially.
- $(\text{EMP}_{\leftarrow})$  Then  $S' = \emptyset$  and  $\emptyset \subseteq S$  trivially.

*Induction* ( $\text{height}(T)>0$ ). The inference rule applied at the root of  $T$  can be either  $(\text{PR}_{\rightarrow})$ ,  $(\text{PR}_{\leftarrow})$ ,  $(\text{CL}_{\rightarrow})$ ,  $(\text{CL}_{\leftarrow})$ ,  $(\text{NEG}_{\rightarrow})$  or,  $(\text{NEG}_{\leftarrow})$ . We check each possibility:

- $(\text{PR}_{\rightarrow})$  or  $(\text{PR}_{\leftarrow})$ . Then  $l(\bar{t}_n)$  is of the form  $p(\bar{t}_n)$  for some predicate  $p$ , and the corresponding inference step must be either of the form

$$\frac{p_1(\bar{t}_n) \rightarrow S'_1 \dots p_k(\bar{t}_n) \rightarrow S'_k}{p(\bar{t}_n) \rightarrow S'} \quad \text{with } S' \supseteq \bigcup_{i=1}^k S'_i$$

or

$$\frac{p_1(\bar{t}_n) \leftarrow S'_1 \dots p_k(\bar{t}_n) \leftarrow S'_k}{p(\bar{t}_n) \leftarrow S'} \quad \text{with } S' \subseteq \bigcup_{i=1}^k S'_i$$

A proof of  $\mathcal{P} \vdash_{\text{SOCLP}} p(\bar{t}_n) \leftrightarrow S$  must also be of the form

$$\frac{p_1(\bar{t}_n) \leftrightarrow S_1 \dots p_k(\bar{t}_n) \leftrightarrow S_k}{p(\bar{t}_n) \leftrightarrow S} \quad \text{with } S = \bigcup_{i=1}^k S_i$$

The proof tree  $T_i$  of  $\mathcal{P} \vdash_{\text{SOCLP}_\rightarrow} p_i(\bar{t}_n) \rightarrow S'_i$  or of  $\mathcal{P} \vdash_{\text{SOCLP}_\leftarrow} p_i(\bar{t}_n) \leftarrow S'_i$  verifies  $\text{height}(T_i) < \text{height}(T)$ , and applying the induction hypothesis it follows that:

- $S_i \subseteq S'_i$  (if  $T$  be a proof tree for  $p(\bar{t}_n) \rightarrow S'$ ) and hence  $S \subseteq S'$
- $S'_i \subseteq S_i$  (if  $T$  be a proof tree for  $p(\bar{t}_n) \leftarrow S'$ ) and hence  $S' \subseteq S$
- (NEG $_{\rightarrow}$ ) or (NEG $_{\leftarrow}$ ). Then  $l(\bar{t}_n)$  is of the form  $\neg p(\bar{t}_n)$  for some predicate  $p$ , and the corresponding inference step must be of the form

$$\frac{p(\bar{t}_n) \rightarrow S'_1}{\neg p(\bar{t}_n) \leftarrow S'} \quad \text{with } S' \subseteq U_n \setminus S'_1$$

or

$$\frac{p(\bar{t}_n) \leftarrow S'_2}{\neg p(\bar{t}_n) \rightarrow S'} \quad \text{with } S' = U_n \setminus S'_2$$

A proof of  $\mathcal{P} \vdash_{\text{SOCLP}} \neg p(\bar{t}_n) \leftrightarrow S$  must be of the form

$$\frac{p(\bar{t}_n) \leftrightarrow S_1}{\neg p(\bar{t}_n) \leftrightarrow S} \quad \text{with } S = U_n \setminus S_1$$

The proof tree  $T'$  of  $p(\bar{t}_n) \rightarrow S'_1$  or  $p(\bar{t}_n) \leftarrow S'_2$  verifies  $\text{height}(T') < \text{height}(T)$ , and applying the induction hypothesis it follows that:

- $S_1 \subseteq S'_1$  (if the inference rule applied is (NEG $_{\leftarrow}$ )), then  $S' \subseteq U_n \setminus S'_1 \subseteq U_n \setminus S_1 = S$  and hence,  $S' \subseteq S$
- $S'_2 \subseteq S_1$  (if the inference rule applied is (NEG $_{\rightarrow}$ )), then  $S = U_n \setminus S_1 \subseteq U_n \setminus S'_2 = S'$  and hence  $S \subseteq S'$

- $(CL_{\rightarrow})$  or  $(CL_{\leftarrow})$ . The proof is similar to the previous cases. Notice from the two rules that the final set  $S$  is obtained from the sets obtained in the premises. Since by induction hypothesis these sets verify the corresponding inclusion the same will occur with the final set  $S$ .

□

Therefore, given  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \leftarrow S_1$ ,  $\mathcal{P} \vdash_{\text{SOCLP}_{\rightarrow}} l(\bar{t}_n) \rightarrow S_2$  and  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S$ , the two sets  $S_1, S_2$  provide, respectively, lower and upper bounds on the set  $S$  (considering  $\subseteq$  as an ordering between sets). Thus, the set  $S_1$  is correct in the sense that contains only answers of the goal, but not necessarily complete, while  $S_2$  is always complete but may be incorrect, since it can include more tuples than those of the answer set. The situation of proposition 3 is, in this sense, the limit case, when the inclusions  $S_1 \subseteq S \subseteq S_2$  become equalities.

A straightforward corollary of Proposition 4 is that for every goal  $l(\bar{t}_n)$  and sets  $S, S_1$  and  $S_2$  such that  $\mathcal{P} \vdash_{\text{SOCLP}} l(\bar{t}_n) \leftrightarrow S$ ,  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \leftarrow S_1$ , and  $\mathcal{P} \vdash_{\text{SOCLP}_{\rightarrow}} l(\bar{t}_n) \rightarrow S_2$ ,  $S_1$  is less or equal than  $S_2$ , i.e.,  $S_1 \subseteq S_2$ . The following proposition shows that this happens even after removing the premise of the existence of a proof for  $l(\bar{t}_n) \leftrightarrow S$ :

**Proposition 5** *Let  $\mathcal{P}$  be a program,  $l(\bar{t}_n)$  a literal with arity  $n$ , and  $S_1, S_2$  two sets such that  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \leftarrow S_1$  and  $\mathcal{P} \vdash_{\text{SOCLP}_{\rightarrow}} l(\bar{t}_n) \rightarrow S_2$ . Then,  $S_1 \subseteq S_2$ .*

**Proof.** Let  $T$  be a proof tree for  $\mathcal{P} \vdash_{\text{SOCLP}_{\leftarrow}} l(\bar{t}_n) \leftarrow S_1$ . The result can be proved using induction on the height of  $T$ .

*Basis* ( $\text{height}(T)=0$ ). The inference rules of the calculus applied at the root can be either  $(\text{TR}_{\leftarrow})$  or  $(\text{EMP}_{\leftarrow})$ :

- $(\text{TR}_{\leftarrow})$ . In the first case  $l(\bar{t}_n)$  is *true* and the only rule that can be applied for proving  $\text{true} \rightarrow S_2$  is  $(\text{TR}_{\rightarrow})$ . Therefore  $S_1 = S_2 = \{()\}$  and the result holds.
- $(\text{EMP}_{\leftarrow})$ : Then  $S_1 = \emptyset$  and the result holds trivially.

*Induction* ( $\text{height}(T) > 0$ ). The inference rules applied at the root of  $T$  can be  $(\text{NEG}_{\leftarrow})$ ,  $(\text{PR}_{\leftarrow})$  or  $(\text{CL}_{\leftarrow})$ .

- $(\text{NEG}_{\leftarrow})$ .  $l(\bar{t}_n)$  must be  $\neg p(\bar{t}_n)$  for some predicate  $p$ , and the corresponding inference step must be of the form

$$\frac{p(\bar{t}_n) \rightarrow S'_1}{\neg p(\bar{t}_n) \leftarrow S_1} \quad \text{with } S_1 \subseteq U_n \setminus S'_1$$

Any proof of  $\mathcal{P} \vdash_{\text{SOCLP}_\leftarrow} p(\bar{t}_n) \rightarrow S_2$  must be of the form

$$\frac{p(\bar{t}_n) \leftarrow S'_2}{\neg p(\bar{t}_n) \rightarrow S_2} \quad \text{with } S_2 = U_n \setminus S'_2$$

The proof tree  $T'$  of the premise  $p(\bar{t}_n) \rightarrow S'_1$  verifies  $\text{height}(T') < \text{height}(T)$ , and applying the induction hypothesis it follows that  $S'_2 \subseteq S'_1$  and hence

$$S_1 \subseteq U_n \setminus S'_1 \subseteq U_n \setminus S'_2 = S_2$$

- (PR $_{\leftarrow}$ ).  $l(\bar{t}_n)$  is  $p(\bar{t}_n)$  for same predicate  $p$  and the corresponding inference step must be of the form

$$\frac{p_1(\bar{t}_n) \leftarrow S_1^1 \dots p_k(\bar{t}_n) \leftarrow S_k^1}{p(\bar{t}_n) \leftarrow S_1} \quad \text{with } S_1 \subseteq \bigcup_{i=1}^k S_i^1$$

Any proof of  $\mathcal{P} \vdash_{\text{SOCLP}_\leftarrow} p(\bar{t}_n) \rightarrow S_2$  must be of the form

$$\frac{p_1(\bar{t}_n) \rightarrow S_1^2 \dots p_k(\bar{t}_n) \rightarrow S_k^2}{p(\bar{t}_n) \rightarrow S_2} \quad \text{with } S_2 \supseteq \bigcup_{i=1}^k S_i^2$$

The proof trees  $T_i$  of the premises  $p_i(\bar{t}_n) \leftarrow S_i^1$  with  $i = 1 \dots k$  verify that  $\text{height}(T_i) < \text{height}(T)$ , and applying the induction hypothesis it follows that  $S_i^1 \subseteq S_i^2$  and hence

$$S_1 \subseteq \bigcup_{i=1}^k S_i^1 \subseteq \bigcup_{i=1}^k S_i^2 \subseteq S_2$$

- (CL $_{\leftarrow}$ ).  $l(\bar{t}_n)$  must be  $p_i(\bar{t}_n)$  with  $p_i$  a program clause:

$$p_i(\bar{a}_n) : -l_1(\bar{b}_{k_1}^1), \dots, l_m(\bar{b}_{k_m}^m)$$

and  $\theta = m.g.u((\bar{a}_n), (\bar{t}_n))$ . The inference step must be of the form

$$\frac{l_1(\bar{b}_{k_1}^1)\theta \leftarrow S_1^1 \dots l_m(\bar{b}_{k_m}^m)\theta \leftarrow S_m^1}{p_i(\bar{t}_n) \leftarrow S_1}$$

Any proof of  $\mathcal{P} \vdash_{\text{SOCLP}_\leftarrow} p(\bar{t}_n) \rightarrow S_2$  must be of the form

$$\frac{l_1(\bar{b}_{k_1}^1)\theta \rightarrow S_1^2 \dots l_m(\bar{b}_{k_m}^m)\theta \rightarrow S_m^2}{p_i(\bar{t}_n) \rightarrow S_2}$$

Applying the induction hypothesis it follows  $S_i^1 \subseteq S_i^2$  for  $i = 1 \dots m$ . Let  $(t'_n)$  be an element of  $S_1$ . By the conditions of the  $(CL_{\leftarrow})$  inference rule, there exists a substitution  $\theta'$  s.t.  $(t'_n) = (t_n)\theta\theta'$  and that  $(\bar{b}_{k_i}^i)\theta\theta' \in S_i^1$  for every  $i = 1 \dots m$ . Then  $(\bar{b}_{k_i}^i)\theta\theta' \in S_i^2$  (since  $S_i^1 \subseteq S_i^2$ ), and by the conditions of  $(CL_{\leftarrow})$  this means that  $(t'_n)$  is an element  $S_2$ , and hence  $S_1 \subseteq S_2$ .

□

The following and last proposition defines the relationship between positive and negative atoms in both calculi

**Proposition 6** *Let  $\mathcal{P}$  be a program,  $p$  a predicate with arity  $n$ , and  $S_1, S_2$  two sets. The two following statements hold:*

- i) *If  $\mathcal{P} \vdash_{SOCLP_{\leftarrow}} p(\bar{t}_n) \leftarrow S_1$  and  $\mathcal{P} \vdash_{SOCLP_{\leftarrow}} \neg p(\bar{t}_n) \leftarrow S_2$ , then  $S_1 \cap S_2 = \emptyset$ .*
- ii) *If  $\mathcal{P} \vdash_{SOCLP_{\leftarrow}} p(\bar{t}_n) \rightarrow S_1$  and  $\mathcal{P} \vdash_{SOCLP_{\leftarrow}} \neg p(\bar{t}_n) \rightarrow S_2$ , then  $S_1 \cup S_2 = U_n$ .*

**Proof.**

- i) Any proof of  $\mathcal{P} \vdash_{SOCLP_{\leftarrow}} \neg p(\bar{t}_n) \leftarrow S_2$  must be of the form

$$\frac{p(\bar{t}_n) \rightarrow S_3}{\neg p(\bar{t}_n) \leftarrow S_2} \quad \text{with } S_2 \subseteq U_n \setminus S_3$$

and therefore  $S_2 \cap S_3 = \emptyset$ . Since  $\mathcal{P} \vdash_{SOCLP} p(\bar{t}_n) \leftarrow S_1$ , by Proposition 5, it follows that  $S_1 \subseteq S_3$  and hence  $S_1 \cap S_2 = \emptyset$ .

- ii) Any proof of  $\mathcal{P} \vdash_{SOCLP_{\leftarrow}} \neg p(\bar{t}_n) \rightarrow S_2$  must be of the form

$$\frac{p(\bar{t}_n) \leftarrow S_3}{\neg p(\bar{t}_n) \rightarrow S_2} \quad \text{with } S_2 = U_n \setminus S_3$$

and therefore  $S_2 \cup S_3 = U_n$ . Since  $\mathcal{P} \vdash_{SOCLP_{\leftarrow}} p(\bar{t}_n) \rightarrow S_1$ , by Proposition 5, it follows that  $S_3 \subseteq S_1$ , and hence  $S_1 \cup S_2 = U_n$ .

□

The two statements of the proposition point out the different and complementary nature of the two calculi: in  $\text{SOCLP}_{\leftarrow}$ , a term cannot belong simultaneously to the sets associated by the calculus to a literal and to its negation, while this is otherwise possible in  $\text{SOCLP}_{\rightarrow}$ . Conversely,  $\text{SOCLP}_{\leftarrow}$  does not ensure that every term is either in the set associated to an atom or in the set of its negation, while  $\text{SOCLP}_{\rightarrow}$  always satisfies this completeness property.

# Chapter 5

## Conclusions and Future Work

The usual operational mechanisms used in logic programming implementations successively yield the answers for a given goal. While this is a good approach in practical implementations, from the point of view of semantics it is worth considering the set of answers of a goal as a whole. The SOCLP calculus presented in this paper represents this point of view. The calculus derivations can be seen as proof trees proving that a set includes all the possible ground answers for a given goal with respect to a logic program. In Theorem 1, we have proved that SOCLP proofs represent faithfully the answer set of a given goal. The main limitation of this calculus is that SOCLP proofs do not always exist; we have only proved completeness (Proposition 2) with respect to the class of hierarchical programs over finite Herbrand universes. Nevertheless, SOCLP proofs are often possible in more general programs and it is part of our future work to extend the completeness result to a broader class of programs. In order to (partially) overcome this limitation, the paper also introduces a calculus  $\text{SOCLP}_{\leftarrow}$  intended for proving subsets of the answer set of a goal.  $\text{SOCLP}_{\leftarrow}$  is coupled through the rule for negation with  $\text{SOCLP}_{\rightarrow}$ , a third calculus which can be used for proving that a given set is a superset of the goal's answer set. The paper proves some basic properties of the three calculi, highlighting that, together, they constitute an useful tool for working with answer sets in logic programming.

As future work, we plan to define a calculus based on SOCLP for defining the semantics of deductive databases [13] and, in particular, of Datalog programs. Notice that SOCLP has already several features that already fit in the usual framework of Datalog, such as the notion of sets of ground tuples as natural answers, and the closed world assumption as a basis for the treatment of negation [2]. Also, the requirement of a finite Herbrand universe for completeness is a condition for Datalog programs, in which no function symbols

are allowed. Thus, the future extension of SOCLP should keep these features while extending the class of complete programs to include non-hierarchical programs, which is the case of Datalog.

**Acknowledgments** The authors are grateful to F.J. López-Fraguas for his interesting and helpful comments and suggestions.

# Bibliography

- [1] Apt, K. R., “From Logic Programming to Prolog,” Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.
- [2] Bidoit, N., *Negation in rule-based database languages: a survey*, in: *Selected papers of the workshop on Deductive database theory* (1991), pp. 3–83.
- [3] Clark, K. L., *Negation as Failure*, in: H. Gallaire and J. Minker, editors, *Logic and Databases* (1987), pp. 293–322.
- [4] Emden, M. H. V. and R. A. Kowalski, *The Semantics of Predicate Logic as a Programming Language*, J. ACM **23** (1976), pp. 733–742.
- [5] Jäger, G. and R. F. Stärk, *The Defining Power of Stratified and Hierarchical Logic Programs*, J. of Logic Programming **15** (1993), pp. 55–77.
- [6] Kowalski, R. A., *Predicate Logic as a Programming Language*, in: J. L. Rosenfeld, editor, *Proceedings of the Sixth IFIP Congress (Information Processing 74)*, Stockholm, Sweden, 1974, pp. 569–574.
- [7] Lloyd, J., “Foundations of Logic Programming,” Springer Verlag, 1984.
- [8] Reiter, R., *On Closed World Databases*, Readings in nonmonotonic reasoning (1987), pp. 300–310.
- [9] Robinson, J. A., *A Machine-Oriented Logic Based on the Resolution Principle*, J. ACM **12** (1965), pp. 23–41.
- [10] Shapiro, E., “Algorithmic Program Debugging,” ACM Distinguished Dissertation, MIT Press, 1982.
- [11] Tessier, A. and G. Ferrand, *Declarative Diagnosis in the CLP Scheme*, in: P. Deransart, M. Hermenegildo and J. Małuszynski, editors, *Analysis and Visualization Tools for Constraint Programming*, number 1870 in LNCS, Springer, 2000 pp. 151–174.

- [12] Ullman, J., "Database and Knowledge-Base Systems Vols. I (Classical Database Systems) and II (The New Technologies)," Computer Science Press, 1995.
- [13] Zaniolo, C., S. Ceri, C. Faloutsos, R. T. Snodgrass, V. S. Subrahmanian and R. Zicari, "Advanced Database Systems," Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.